ECE385

Fall 2021

Experiment 2

Data Storage

Ge Yuhao, Lou Haina
D231, Sept.27, 2021
TA: Chenhao Wang

1 Introduction

In this lab, we design and construct a simple 2-bit, 4-word shift-register storage unit. We can do the "Store", "Fetch", "Load" with this unit.

LOAD Storage buffer register (SBR) is loaded with the data from DIN1 and DIN0.

STORE The value in the SBR is stored into the address specified by the storage address register (SAR).

FETCH Read the value in address specified by SAR to SBR.

DO NOTHING The shift will continuously shifting to maintain the original bits.

2 pre-lab writeup

Why it is better not to gate the clock? Because with other input, the output of the gate may encounter a static hazard which makes the clock signal unstable (May have more rising edges in one period). This may make other chips be clocked abnormally.

Written description for circuit operation This circuit accomplished three basic operations for data storage: Load, Store and Fetch. We load new data into SBR(storage buffer register, built by flip-flop) and store it(write) into our register at address read from SAR, then we read the data after we input its address(fetch from shift register into SBR), this whole process has completed.

Operation of controller We use controller with some logic in it to select true operation. For example, LDSBR will control the 3-1 MUX to input new data when it is high and the counter, and comparator contained output high voltage AND with STORE control the 2-1 MUX to write data into register. Also, fetch AND with Comparator to control 3-1 MUX read the data of SAR address.

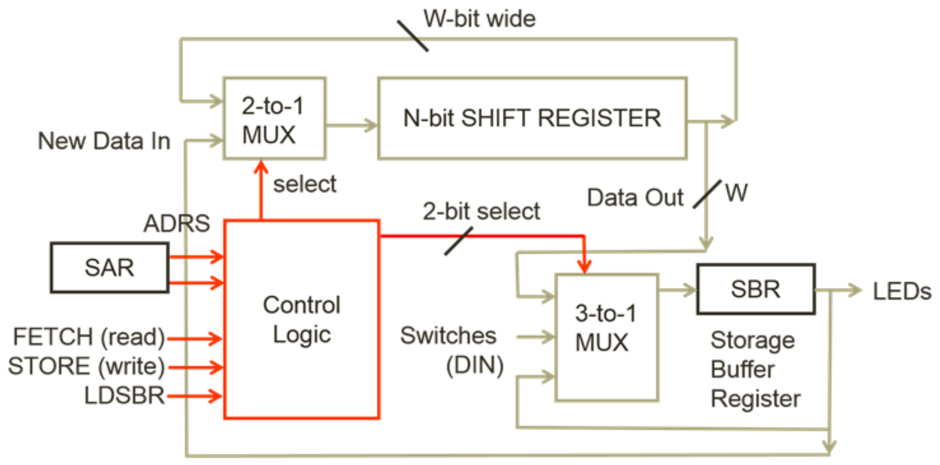

Figure 1: Block Diagram for circuit

3 Operation of the memory circuit

3.1 Describe how the addressing is implemented. When does the circuit commit a read or write from/to the input switches and output register respectively.

- The addressing is implemented through a counter and a comparator. The counter will count from 0 to 3, which has the same period with the shift register. Therefore, there will be a specific counter number for each bit of the shift register. Each time we want to read or write a bit on a specific address, we need to compare the value between the counter and the SAR. Then the read or the write operation can be conducted.

- The selection is implemented through a 3-1 MUX and 2-1 MUX. Together with the control logic, the 2-1 MUX will help to choose whether the left-most bit will store the data from SBR or store back the right-most bit from shift register. The 3-1 MUX will help to choose whether the SBR read from switches, shift register, or itself.

3.2 Describe how a write operation is performed on the memory. Describe what switches you flip and in what order. Describe intuitively how the data flows through the circuit at each clock cycle.

- During a write operation, we do want to replace the old data with new data. To serve both purposes, a 2-to-1 MUX can be placed at the serial input of the shift registers, taking either the new data or the old data depending on the current operation. A counter and a comparator is used to make sure that the leftmost bit of the register is corresponding to the SAR.

- To conduct WRITE, we should set STORE to high and FETCH to low, and set the SAR signal to the address we want to store.

- For the clock cycle before the counter having the same value as SAR, the shift register will keep right shifting, and the rightmost bit will be transferred to the leftmost one through the 2-to-1 MUX. After the counter having the same value as SAR, the value on the leftmost bit of the shift register will be lost and be overwritten by the data from SBR transferred through the 2-to-1 MUX. As for data in the SBR, the 3-to-1 MUX will always make the input of SBR be the original value in it to make sure that the value does not change.

3.3 Describe how a read operation is performed on the memory. Describe what switches you flip and in what order. Describe intuitively how the data flows through the circuit at each clock cycle.

- During a read operation, we do want to replace the old data in the SBR with new data from the shift register. To serve this purpose, a 3-to-1 MUX can be placed at the serial input of the shift registers, taking either the new data from a switches or the register, or the old data in SBR depending on the current operation. A counter and a comparator is used to make sure that the rightmost bit of the register is corresponding to the SAR.

- To conduct READ, we should set STORE to low and FETCH to high, and set the SAR signal to the address we want to read.

- For the clock cycle before the counter having the same value as SAR, the 3-to-1 MUX will always make the input of SBR be the original value in it to make sure that the value does not change. After the counter having the same value as SAR, the value on the rightmost bit of the shift register will be sent to the SBR transferred through the 3-to-1 MUX. As for data in the shift register, the shift register will keep right shifting, and the rightmost bit will be transferred to the leftmost one through the 2-to-1 MUX to maintain the original data.

4 Written description and block diagram of memory circuit implementation

4.1 High-level description

To build a 2-bit, 4-word shift-register storage unit, we use Shift Registers, MUXs, Flip-Flops, Counter, Comparator and combinational logic.

We use a counter and a comparator with an input "SAR" to make sure that the leftmost or the rightmost bit of the shift register is the correct one to write or read each time we want to STORE or FETCH. Only when the values in the counter and SAR are equal can the comparator give a high output.

With the output of the comparator and the "Store" signal, we design an combinational logic to control a 2-1 MUX, which determines the leftmost input of the shift register (Restore the rightmost bit or accept a new one from SBR output).

The two shift registers are used to store the bits we want. Each has a length of 4, and with two we can build a 2-bit, 4-word shift-register storage unit.

We use a D Flip-Flop to temporary store one bit. A 3-1 MUX is used to control the input of the Flip-Flop, which can be the rightmost bit from shift register, or the input from "DIN" signal, or the original bit in Flip-Flop. A combinatinal logic which accepts inputs like "LDSBR", "FETCH", "Counter output" is used to control the 3-1 MUX.

4.2 A high-level block diagram

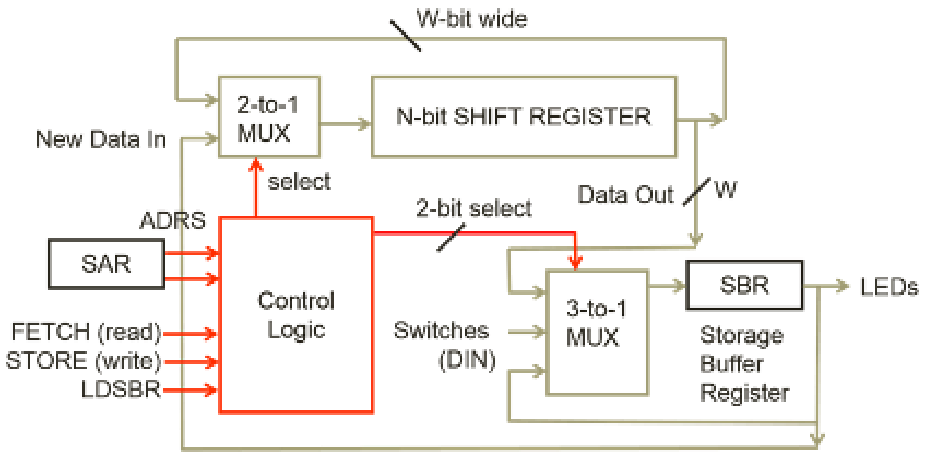

Figure 2: High-level block diagram

5 Control Unit

5.1 Description of the control unit

The control unit accept several inputs: STORE, LDSBR, FETCH, where we assume that only one of them will be high in each moment. The control unit also accept other input: SAR0, SAR1.

The circuit will conduct LOAD when the input LDSBR is high. The control unit will send signal to the 3-1 MUX to load the SBR with DIN0 and DIN1.

The circuit will conduct STORE when the input STORE is high, and the Counter outputs equal to SAR0 and SAR1. The control unit will send signal to the the 2-1 MUX to load the shift register with the value from SBR

The circuit will conduct FETCH when the input FETCH is high, and the Counter outputs equal to SAR0 and SAR1. The control unit will send signal to the the 3-1 MUX to load the SBR with the bit from the shift register.

The circuit will do nothing when both of the three inputs are low. The control unit will send signal to the 3-1 MUX and 2-1 MUX to make the shift register restore the rightmost bit and make the Flip-Flop restore its original value.

5.2 Two block diagrams, one like figure 3 in the lab manual and one containing only the inside of the control unit.

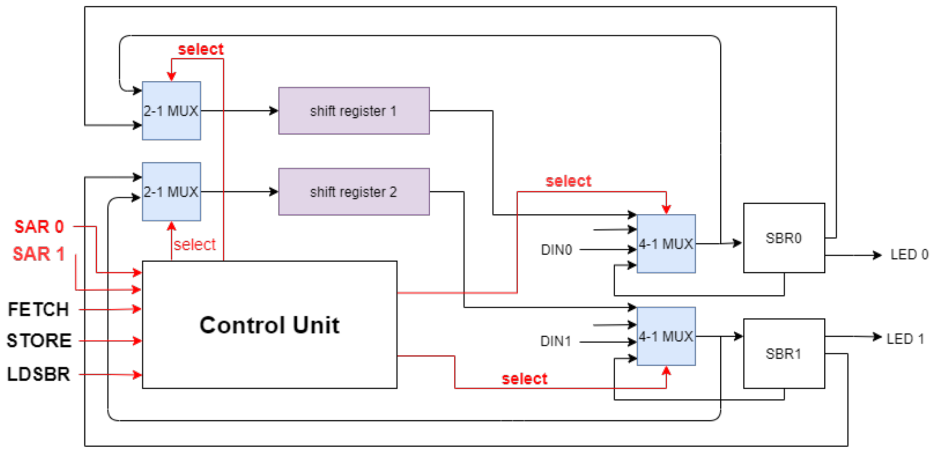

Figure 3: Block Diagram for circuit

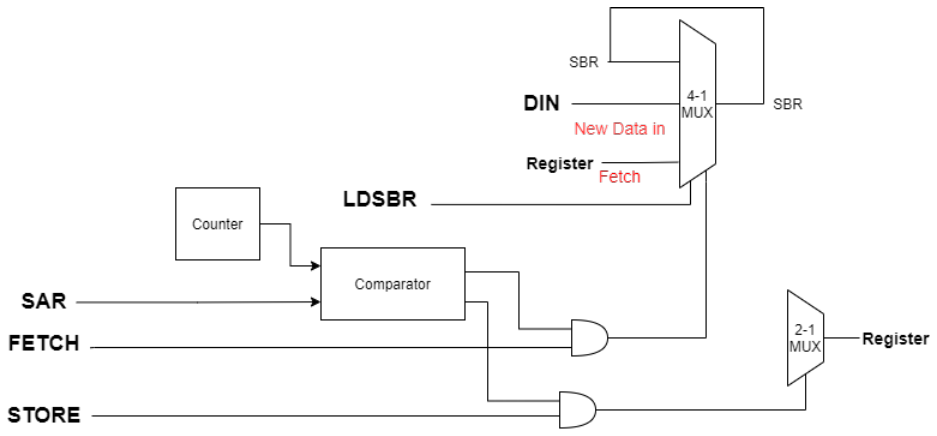

Figure 4: Block Diagram for control unit

6 Design steps taken

Truth tables and K-maps we need to derive the logic for the two select bits of 4-to-1 MUX and the logic for one select bit of 2-to-1 MUX.

LDSBR	FETCH	COMPARE	SEL1	SEL2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	x	x

Table 1: Truth table for 4-1 MUX

Then we construct the following truth table and Karnaugh map to formulize the result.

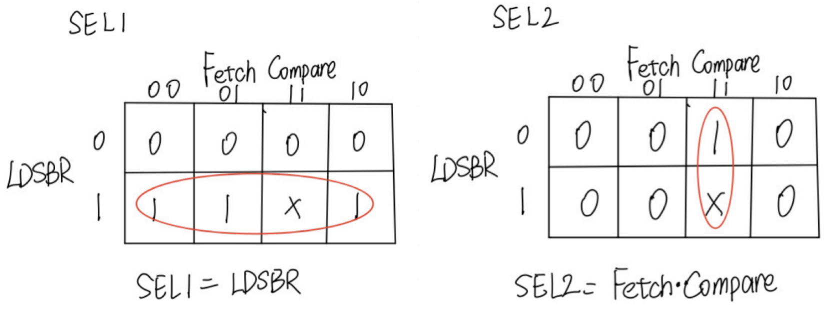

Figure 5: K-maps for 4-1 MUX

The Karnaugh map helps us to derive the formula of SEL1 =LDSBR , SEL2 = FTECH*COMPARE. As for the 2-to-1 MUX, the logic is trivial which can be derived easily: SEL = STORE*COMPARE

7 Component Layout Sheet

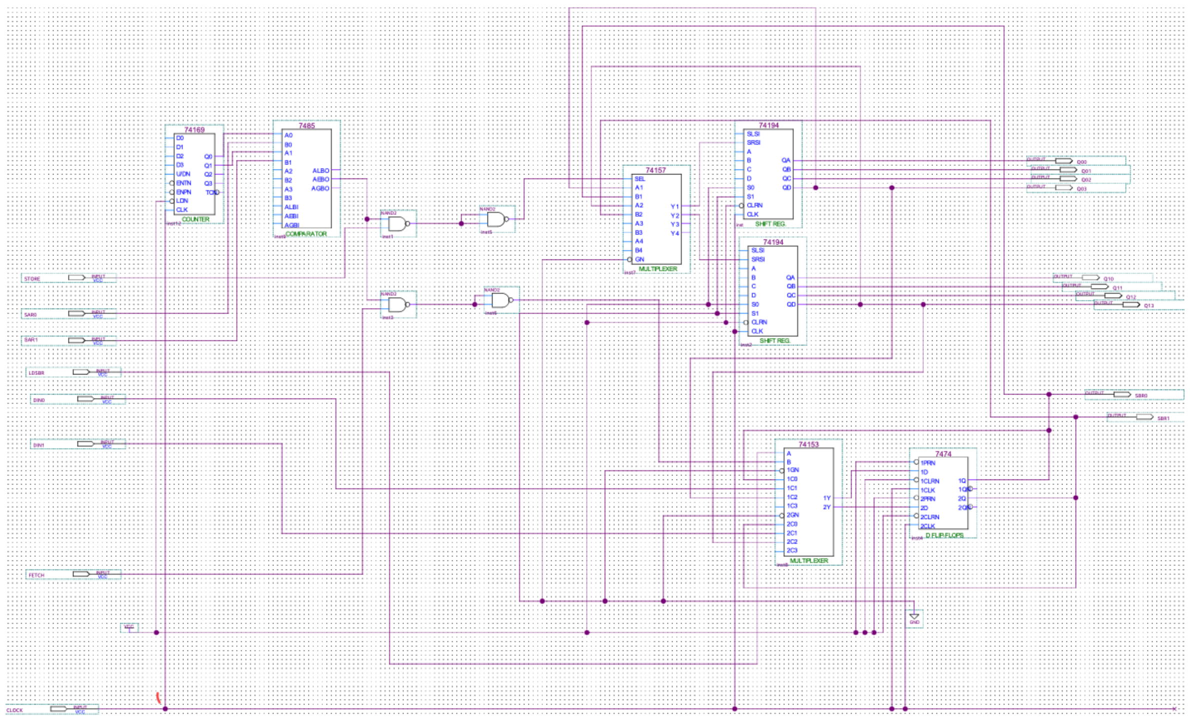

Figure 6: Component Layout Sheet

8 Post-Lab

8.1 What are the performance implications of your shift register memory as compared to a standard SRAM of the same size?

The SRAM use paired inverters to store bits, while the shift registers memory use flip-flops to store bits. Basing on this, there are several differences between them.

- The SRAM has a more complex internal structure which results in a larger space needed, while the shift register need less space.
- The SRAM is static but the data in shift register keeps shifting. So the shift register consumes more energy especially when the data is not being accessed.
- The shift register has a simpler complexity of addressing finding, but it need more time on accessing specific memory location as it needs to wait the needed bit shifting to the leftmost place. For the SRAM with the same size, it can reach the specific bit directly which is faster.

8.2 What are the implications of the different counters and shift register chips, what was your reasoning in choosing the parts you did?

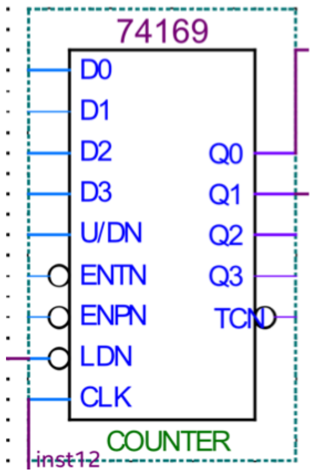

Figure 7: Counter

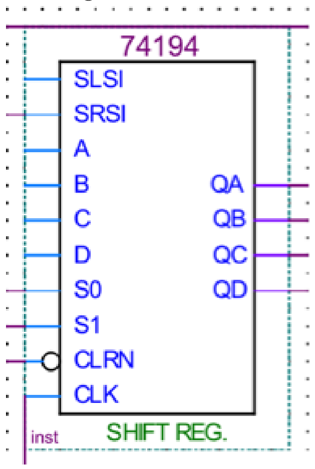

Figure 8: Shift Register

Why we use 74169 as counter? The counter is used to memorize the bits' location in the shift register. Because the register can store up to 4 bits, so we only need to use the 2 lowest bits of the counter, namely Q1 and Q2. We also connect LDN with VCC to make the counter to be counting continuously in keeping with the shift register. We used this 74169 instead of the 7493 Ripple Counter because this counter is synchronous and thus more stable. As the glitches in the ripple counter happen right after every clock edge, which temporarily outputs wrong values before settling on the correct one. Also we can initiallize the value in 74169 while 7493 can't do this.

Why we use 74194 as shift register? The shift register is used to store bits and keep shifting for read and write. As we only need to write the leftmost bit, only SRSI is connected to the output of the MUX. S0 is connected to VCC and S1 is connected to GND to make it always shift to right. 74LS194 is an ideal choice for this lab because it can. Because we only need serial input and we need the clock period is 1 millisecond, we use 74LS194 instead of 74LS95. 74LS95 is a 4 bit parallel-access shift register, and have two clock input which is more complicated.

9 Conclusion

Bugs encountered and corrective measures taken As we have a very careful designing process, the logic is fine during our project. However, as we are not so familiar with the chips and Quartus Prime, we encountered many bugs related to the the chip connection and file path. We overcame those difficulties by looking up the Data-sheet book more carefully and frequently and learning the use of Quartus Prime again. We also meet difficulties in saving our work and set save path for our waveform file, after learn from the instruction video, we run it without error. Apart from this, we also have encountered connection problems of input with MUX selection pin. By writing truth-table and k-map carefully, we connect it correctly.

Summary In this lab, we design and construct a simple 2-bit, 4-word shift-register storage unit. We can do the ”Store”, ”Fetch”, ”Load” with this unit. By doing this lab, we learn much on how to use Quartus Prime to simulate complicated circuit. We learn the structure of many chips like 74194 and 74169, and understand that connecting the right feet is very important. We also have a basic idea of how a synchronized circuit work.